INTRODUCTION TO JAVA

Outline – Module 1 (Recap Java programming fundamentals)

- Introduction to Java
- Overview of JDK/JRE/JVM
- Java Language Constructs
- Object Oriented Programming with Java
- Exception Handling

Outline – Module 2

(Java Collection Framework)

- Java Collection Overview
- Generics Overview

Outline – Module 3 (Java Concurrency and other features)

- Introduction to Multi-Threading
- Java Concurrency API Overview
- Java Annotations Overview
- Java Utils package Overview

Outline – Module 4

(Java 8 Features)

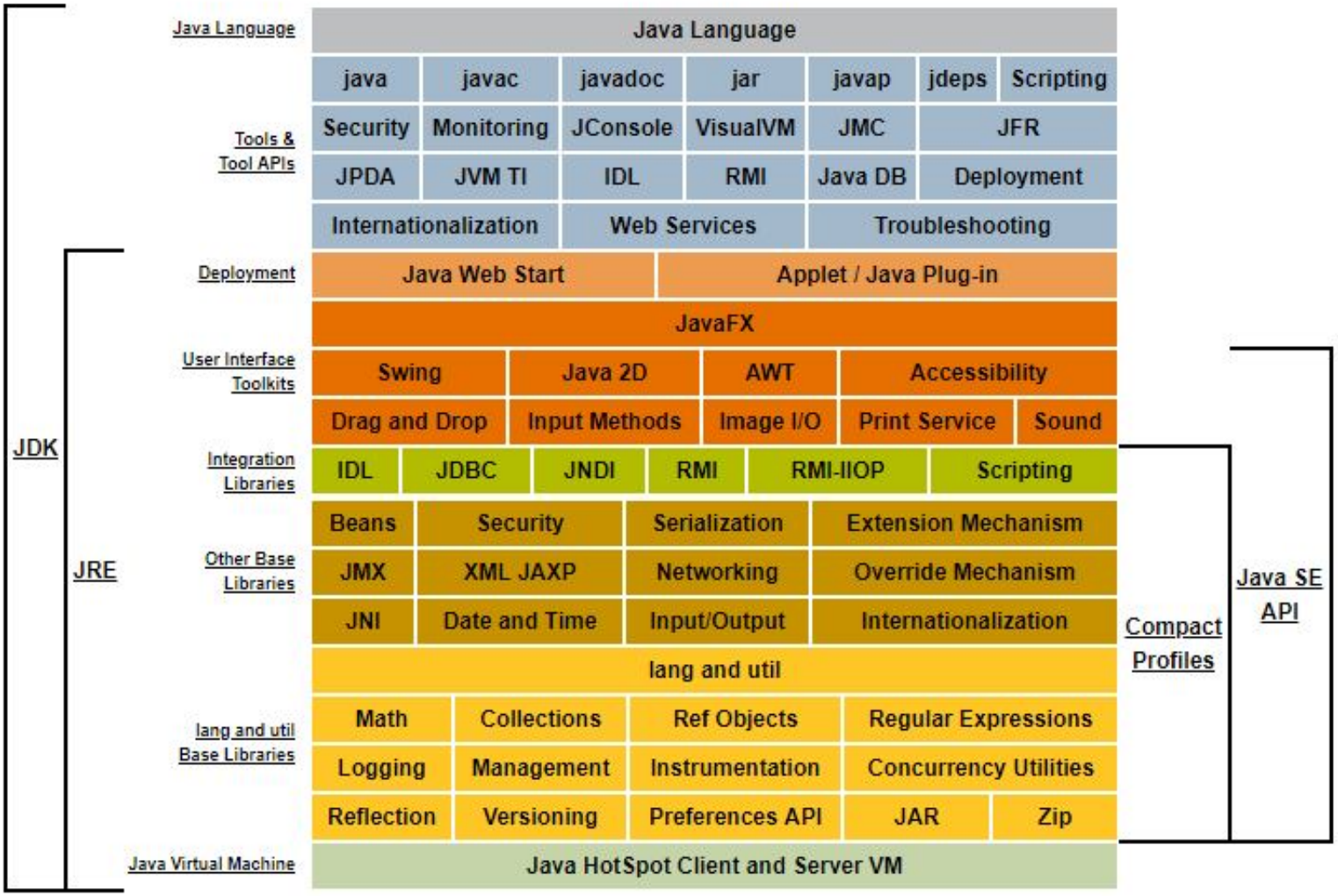
- Fundamentals of Functional Programming
- Lambda Expressions
- Functional Interfaces
- Stream API - foreach, map, filter, parallel processing, collectors, etc.
- Enhanced Collections Framework with Lambdas
- Enhanced Concurrency API with Lambdas

Outline – Module 5

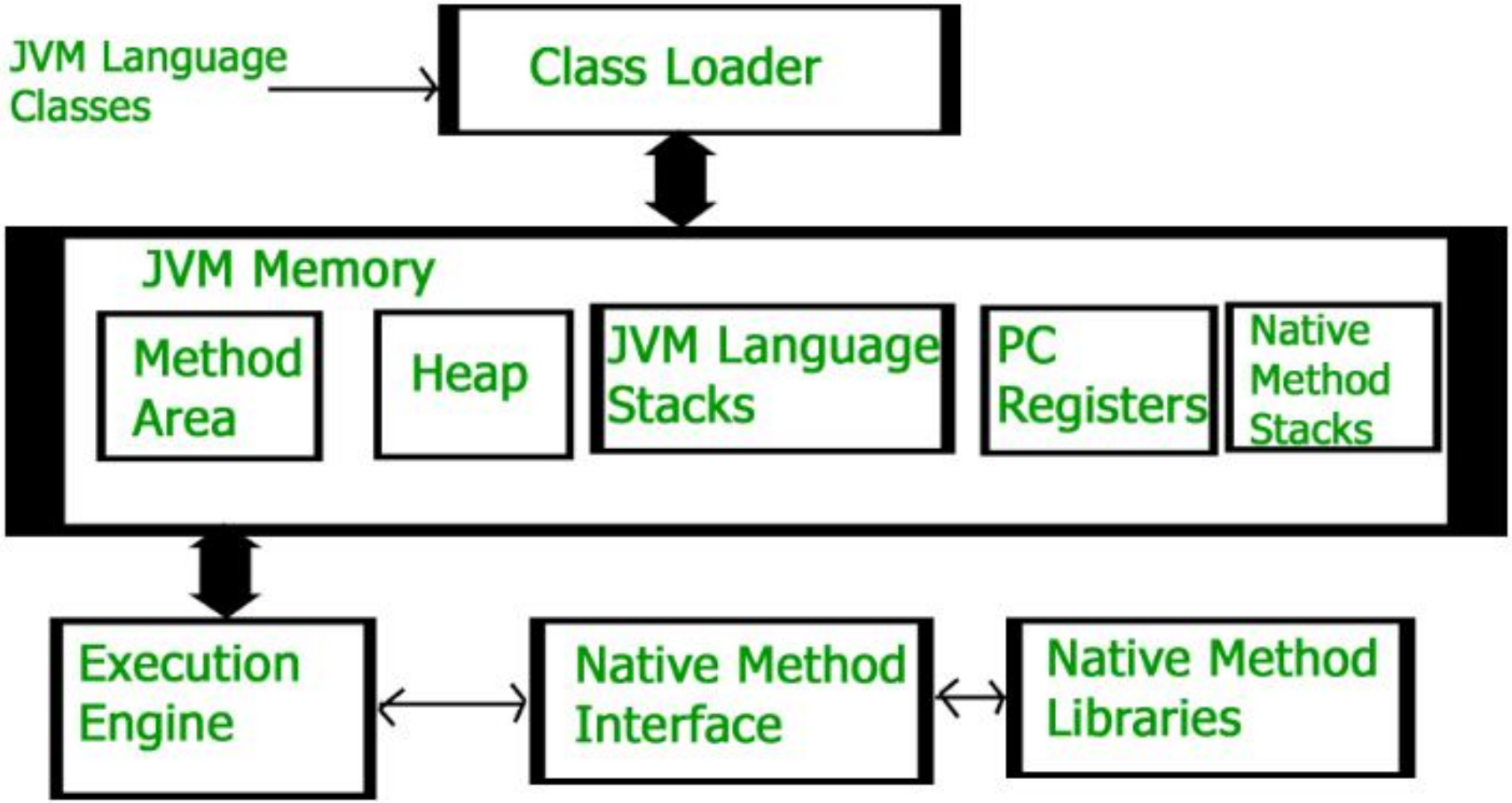
(JDBC Programming)

- Introduction to JDBC
- Loading Driver / Creating Data Source
- Creating Connection
- Preparing/Compiling Statements
- Executing Statements
- Processing ResultSet

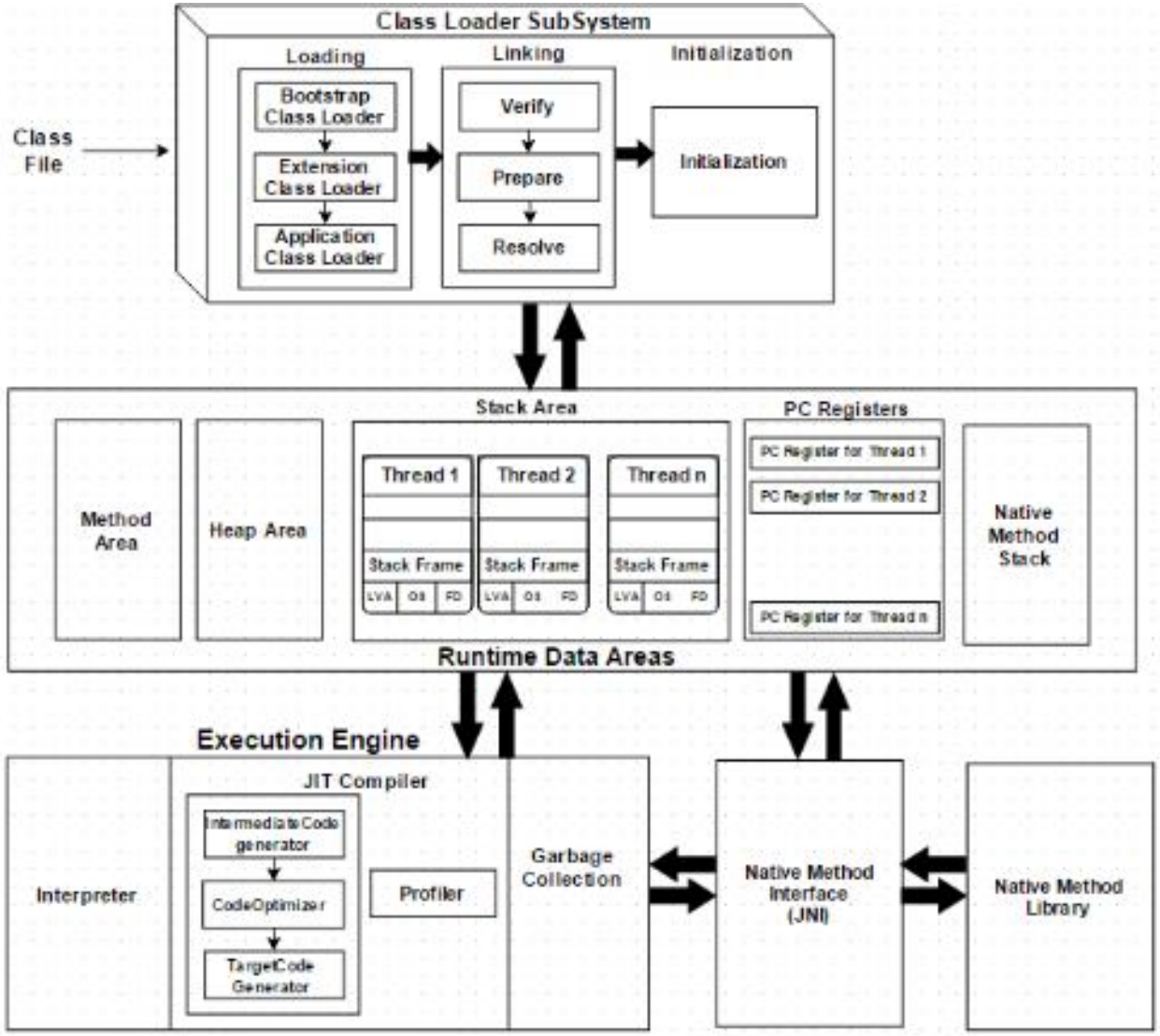
Java Conceptual Model (JVM/JRE/JDK)



JVM Architecture



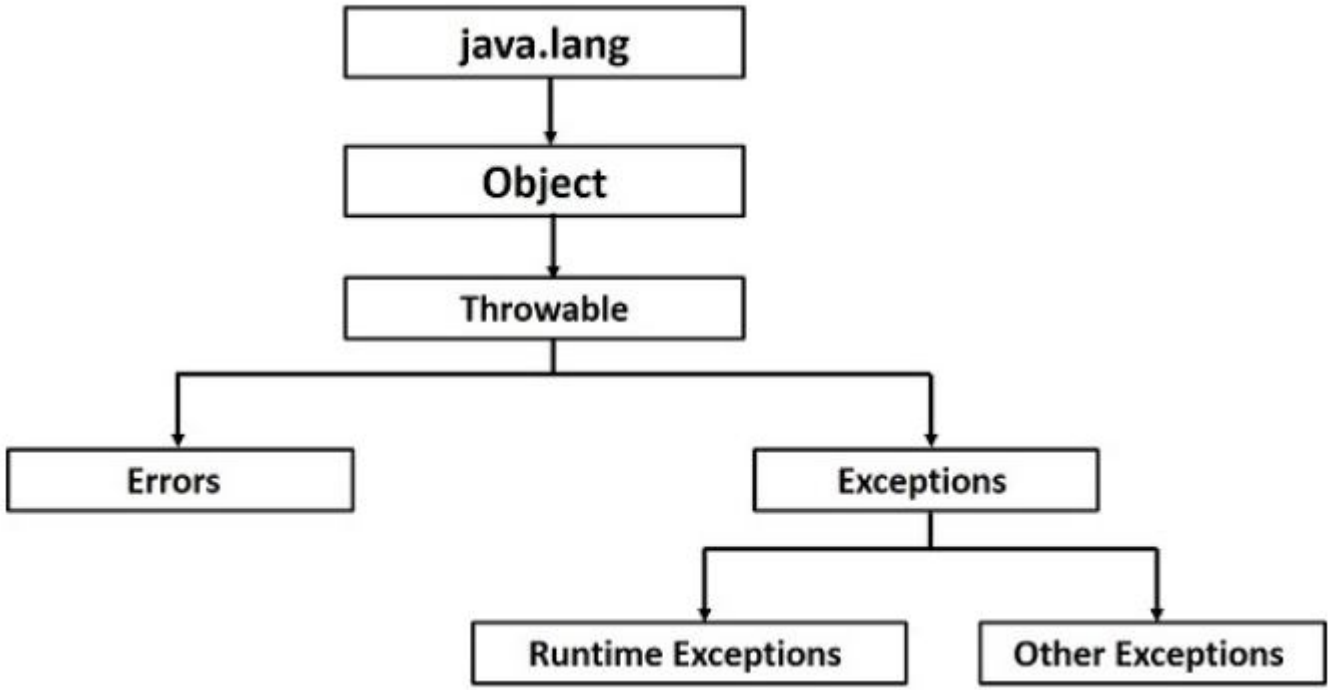
JVM Architecture (detailed)



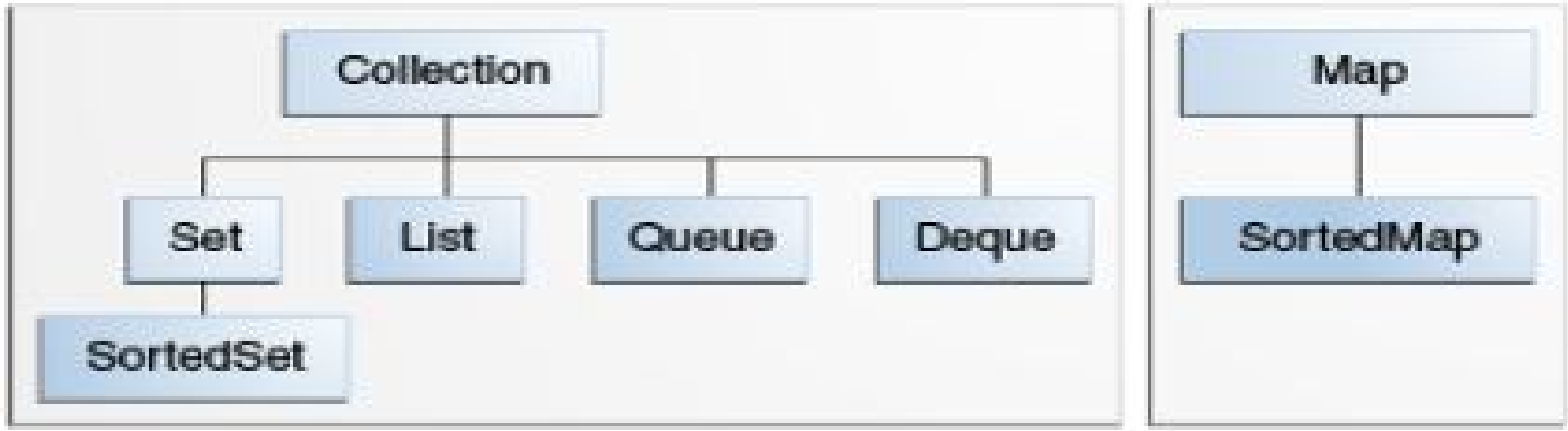
Java Keywords

abstract	default	if	private	this
assert	do	implements	protected	throw
boolean	double	import	public	throws
break	else	instanceof	return	transient
byte	enum	int	short	try
case	extends	interface	static	void
catch	final	long	strictfp	volatile
char	finally	native	super	while
class	float	new	switch	
continue	for	package	synchronized	

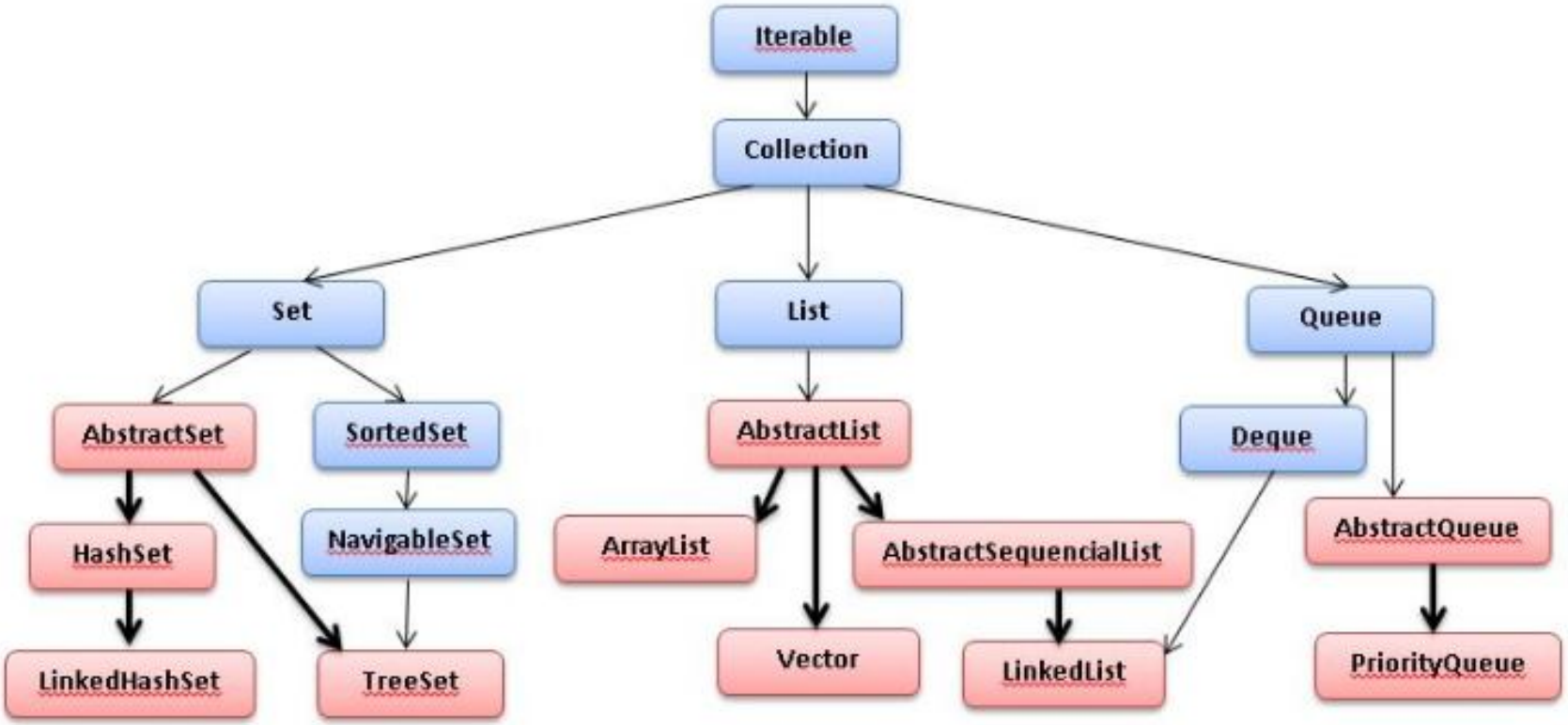
Exception Hierarchy



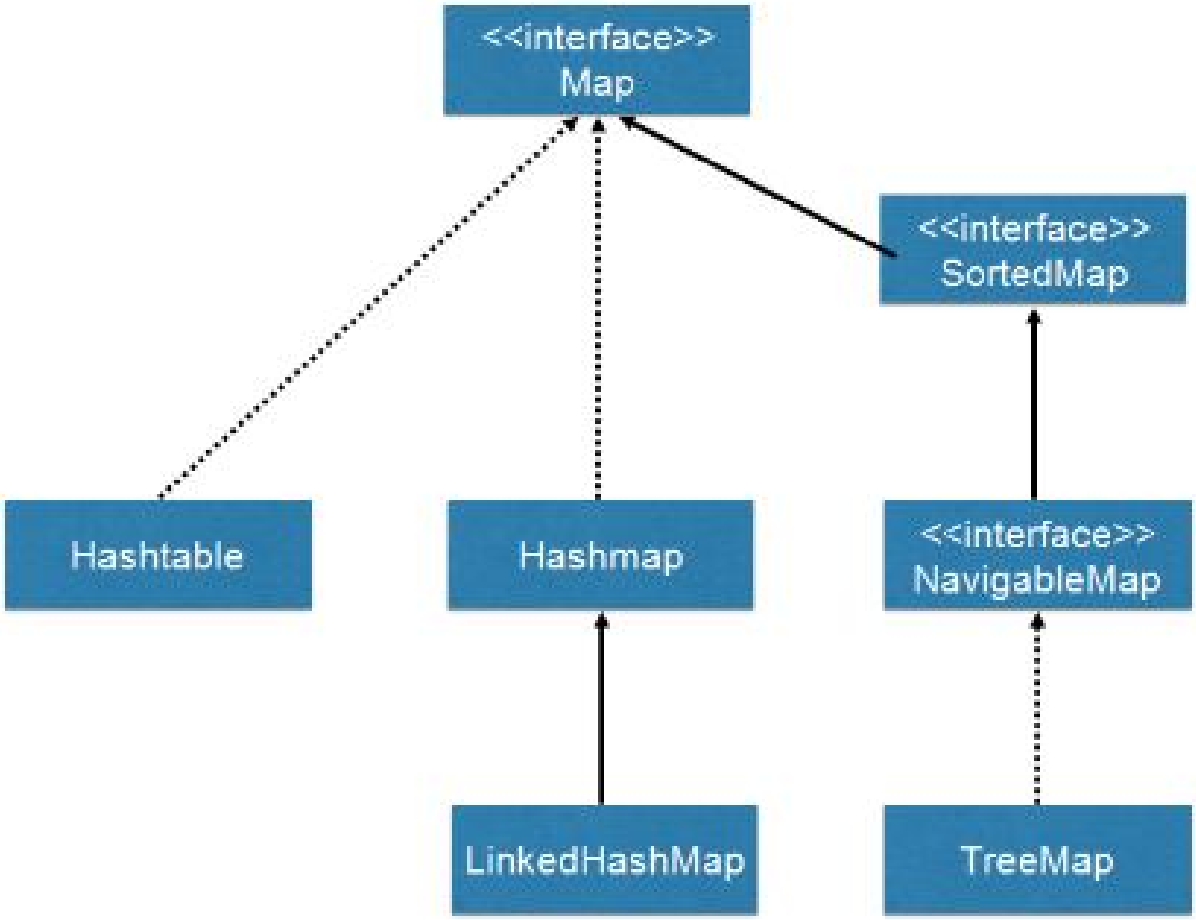
Collection Hierarchy (Interfaces)



Collection Hierarchy



Collection Hierarchy (contd.)

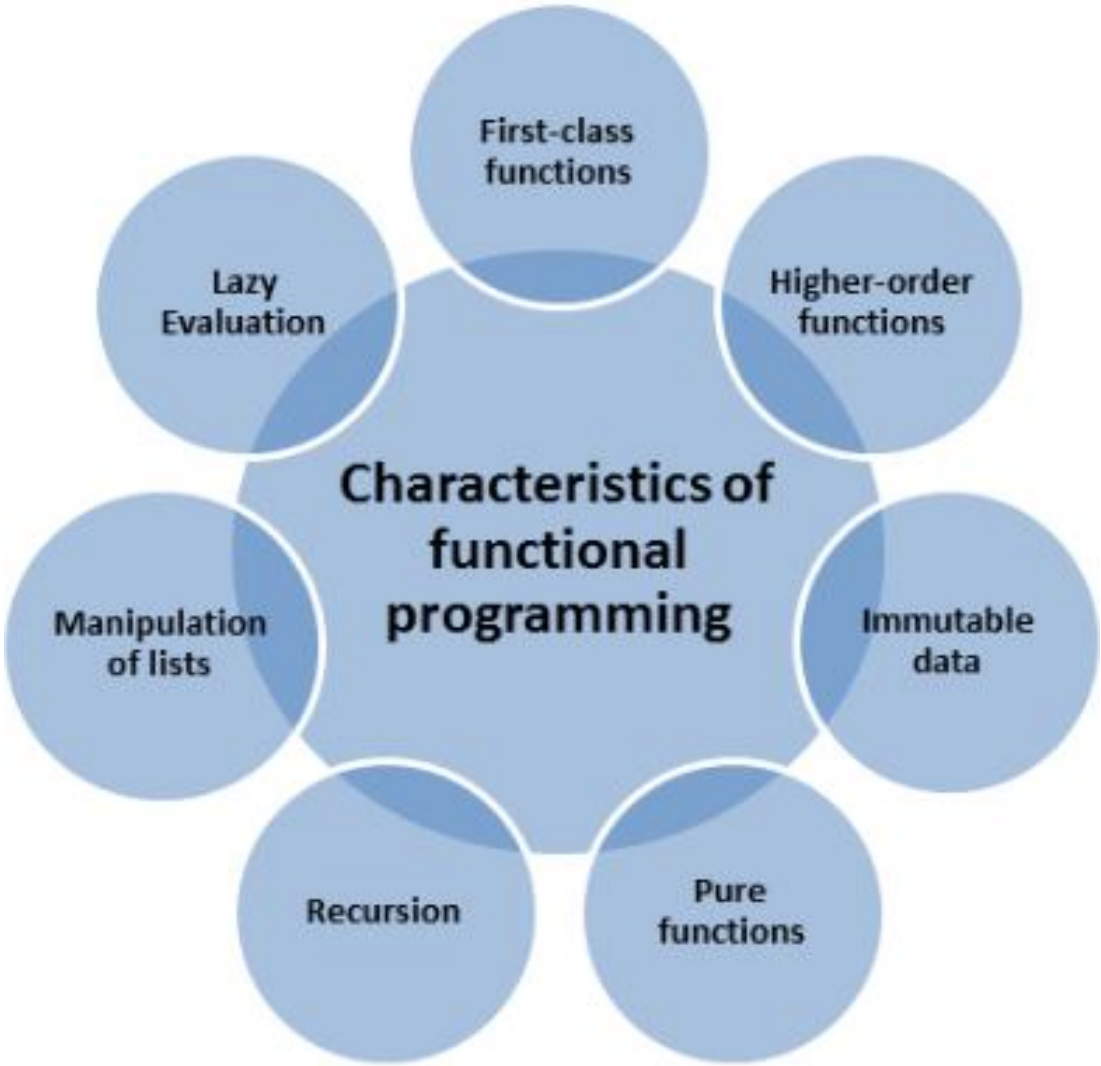


Functional Programming

Functional programming is just a style, is a programming paradigm that treats computation as the evaluation of functions and avoids state and mutable data…

- *"Functions as primary building blocks"* (first-class functions)
- programming with *"immutable"* variables and assignments, no s
  - Programs work by returning values instead of modifying data

Functional Programming Characteristics



Functional vs Object Oriented Programming

Functional Programming	OOP
Uses Immutable data.	Uses Mutable data.
Follows Declarative Programming Model.	Follows Imperative Programming Model.
Focus is on: "What you are doing"	Focus is on "How you are doing"
Supports Parallel Programming	Not suitable for Parallel Programming
Its functions have no-side effects	Its methods can produce serious side effects.
Flow Control is done using function calls & function calls with recursion	Flow control is done using loops and conditional statements.
It uses "Recursion" concept to iterate Collection Data.	It uses "Loop" concept to iterate Collection Data. For example: For-each loop in Java
Execution order of statements is not so important.	Execution order of statements is very important.
Supports both "Abstraction over Data" and "Abstraction over Behavior".	Supports only "Abstraction over Data".

Lamda Expression

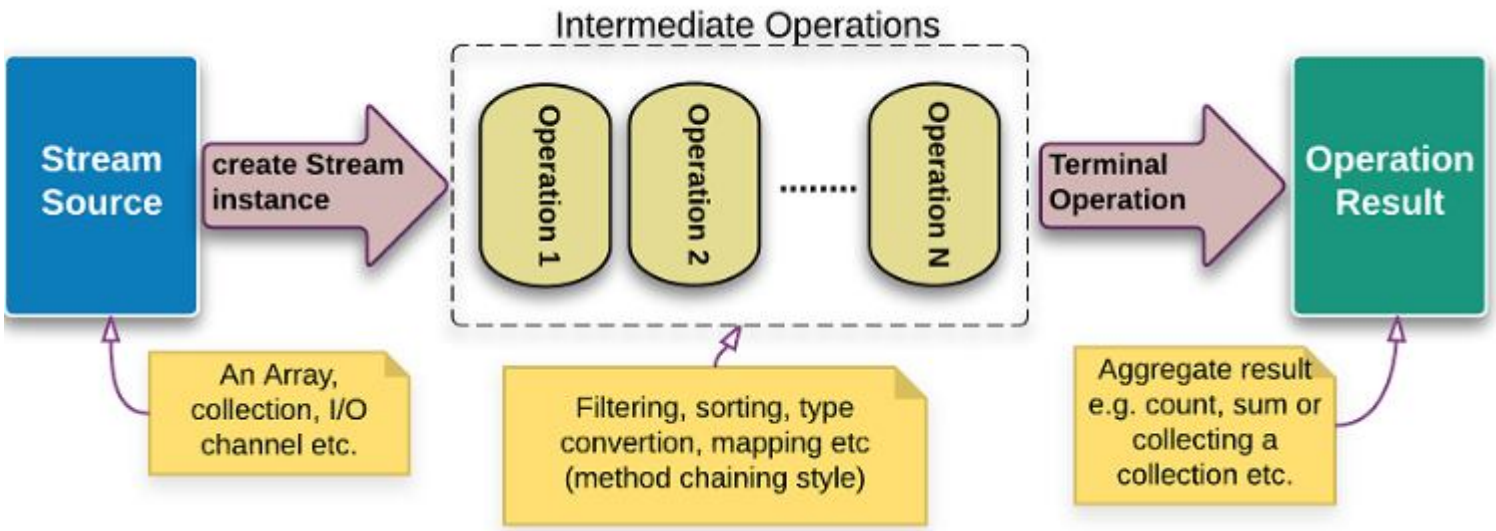
parameter -> expression body

- No arguments:
```
() -> System.out.println("Hello")
```
- One argument:
```
s -> System.out.println(s)
```
- Two arguments:
```
(x, y) -> x + y
```
- With explicit argument types:
```
(Integer x, Integer y) -> x + y
```
- Multiple statements:
```
(x, y) -> {
    System.out.println(x);
    System.out.println(y);
    return (x + y);
}
```

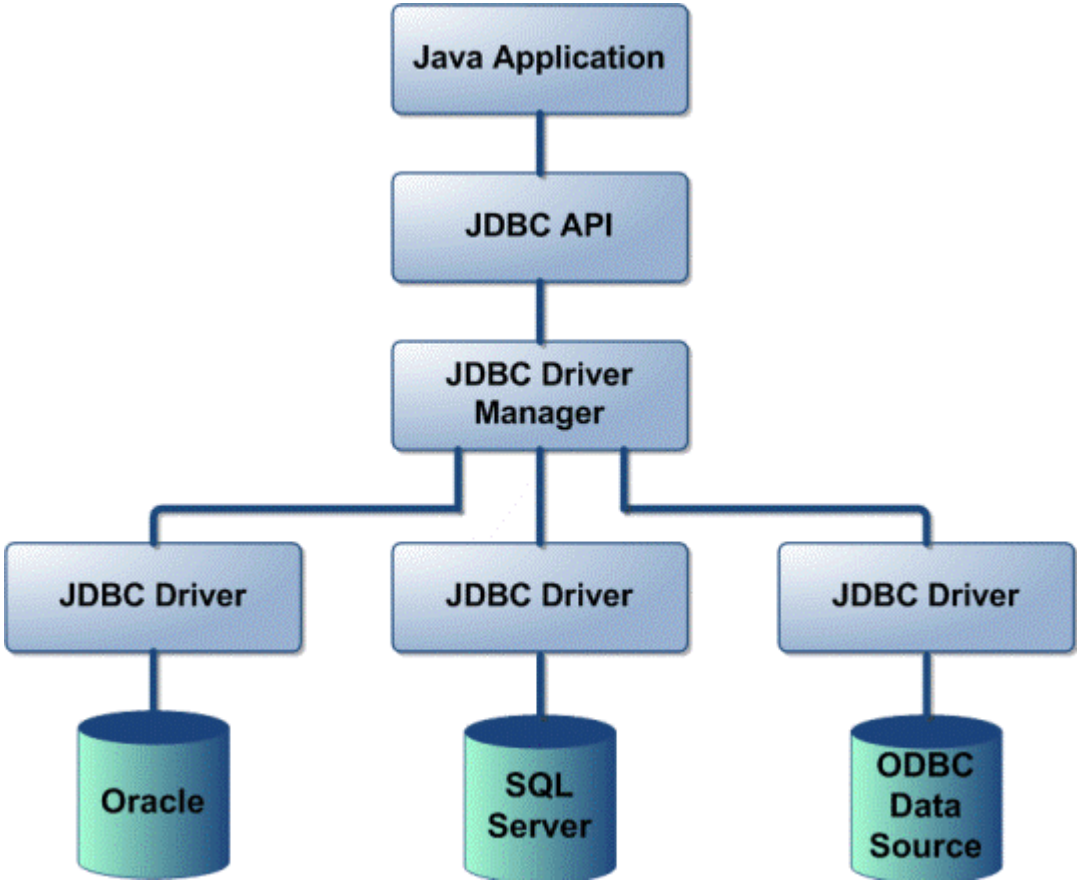
Java 8 Features

- Lamda Expressions
- Functional Interfaces
- Method References
- Default Methods
- Streams

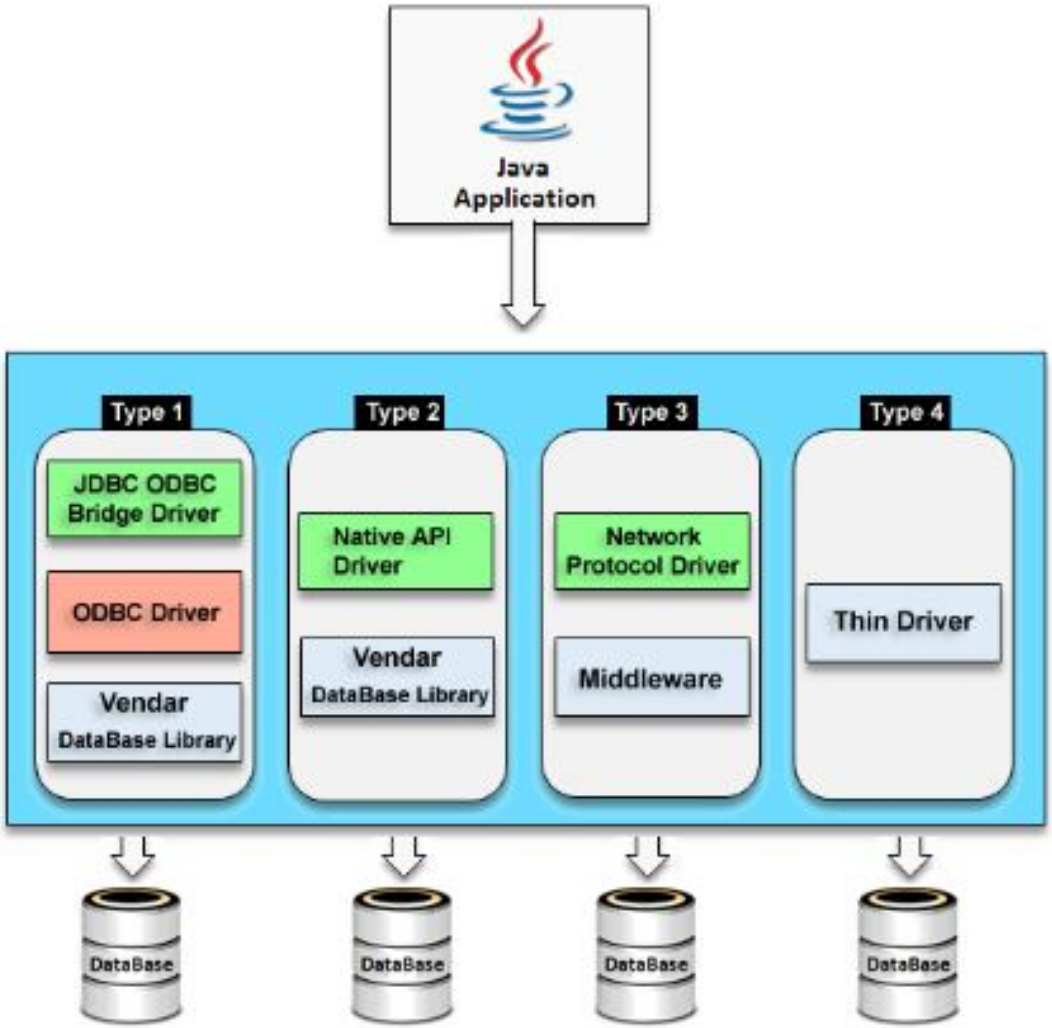
Java Streams Overview



JDBC Overview



JDBC Drivers



Thank You!